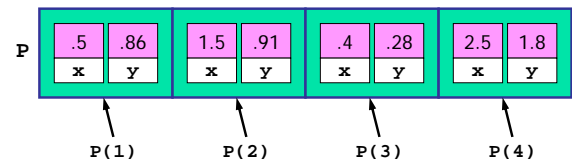


- Previous Lecture:
  - Structure & structure arrays
- Today's Lecture:
  - Structure arrays
  - Working with large data files
  - Built-in `sort` function
  - Read Chapter 11 to learn about the `.bin` file format
- Announcement:
  - P5 due Nov 11 at 11pm
  - Prelim 3 Tuesday, Nov 16th
  - Check website for changes in office/consulting hrs
    - Fan's Wednesday office hr moved to afternoon the rest of this semester
    - No consulting tonight 8-10pm

### Structure Arrays

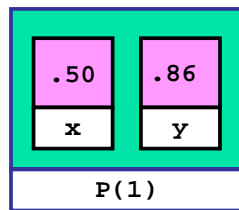
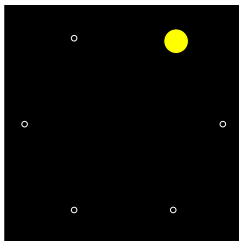
- An array whose components are structures
- All the structures must be the same (have the same fields) in the array
- Example: an array of points (point structures)



Lecture 20

2

### An Array of Points



`P(1) = MakePoint(.50,.86)`

Lecture 20

3

### Function returning an array of points (point structures)

```
function P = CirclePoints(n)
```

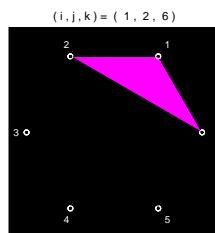
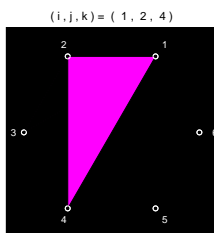
```
theta = 2*pi/n;
for k=1:n
    c = cos(theta*k);
    s = sin(theta*k);
    P(k) = MakePoint(c,s);
end
```

Lecture 20

9

### Example: all possible triangles

- Place `n` points uniformly around the unit circle.
- Draw all possible unique triangles obtained by connecting these points 3-at-a-time.



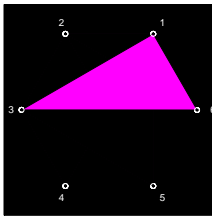
```
function DrawTriangle(P,Q,R,c)
% Draw c-colored triangle;
% triangle vertices are points P,
% Q, and R.
```

```
fill([P.x Q.x R.x], ...
     [P.y Q.y R.y], c)
```

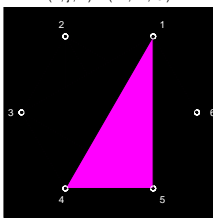
Lecture 20

11

$(i, j, k) = (1, 3, 6)$



$(i, j, k) = (1, 4, 5)$



The following triangles are the same: (1,3,6), (1,6,3), (3,1,6), (3,6,1), (6,1,3), (6,3,1)

Lecture 20
12

Bad!  $i, j$ , and  $k$  should be different, and there should be no duplicates

```

for i=1:n
  for j=1:n
    for k=1:n
      % Draw a triangle with vertices
      %   P(i), P(j), and P(k)
    end
  end
end

```

Lecture 20
13

All possible  $(i,j,k)$  combinations but avoid duplicates.  
Loop index values have this relationship  $i < j < k$

$i \ j \ k$

1	2	3
1	2	4
1	2	5
1	2	6
1	3	4
1	3	5
1	3	6
1	4	5
1	4	6
1	5	6

$i = 1$

2	3	4
2	3	5
2	3	6
2	4	5
2	4	6
2	5	6

$i = 2$

3	4	5
3	4	6
3	5	6

$i = 3$

4	5	6
---	---	---

$i = 4$

```

for i=1:n-2
  for j=i+1:n-1
    for k=j+1:n
      disp([i j k])
    end
  end
end

```

Lecture 20
14

All possible  $(i,j,k)$  combinations but avoid duplicates.  
Loop index values have this relationship  $i < j < k$

```

for i=1:n-2
  for j=i+1:n-1
    for k=j+1:n
      % Draw triangle with
      % vertices P(i),P(j),P(k)
    end
  end
end

```

Lecture 20
16

All possible triangles

```

% Drawing on a black background
for i=1:n-2
  for j=i+1:n-1
    for k=j+1:n
      DrawTriangle( P(i),P(j),P(k),'m')
      DrawPoints(P)
      pause
      DrawTriangle(P(i),P(j),P(k),'k')
    end
  end
end

```

Lecture 20
17

Still get the same result if all three loop indices end with  $n$ ?

$i \ j \ k$

1	2	3
1	2	4
1	2	5
1	2	6
1	3	4
1	3	5
1	3	6
1	4	5
1	4	6
1	5	6

$i = 1$

2	3	4
2	3	5
2	3	6
2	4	5
2	4	6
2	5	6

$i = 2$

3	4	5
3	4	6
3	5	6

$i = 3$

4	5	6
---	---	---

$i = 4$

```

for i=1:n
  for j=i+1:n
    for k=j+1:n
      disp([i j k])
    end
  end
end

```

Lecture 20
19

## Structures with array fields

Let's develop a structure that can be used to represent a colored disk. It has four fields:

**xc:** x-coordinate of center  
**yc:** y-coordinate of center  
**r:** radius  
**c:** rgb color vector

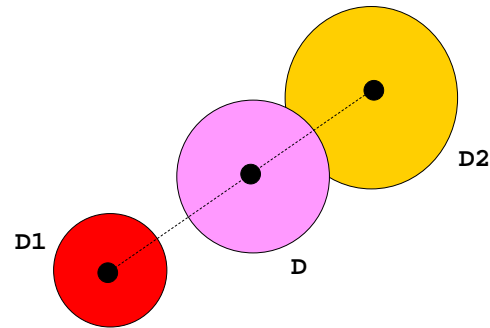
Examples:

```
D1 = struct('xc',1,'yc',2,'r',3,...
           'c',[1 0 1]);
D2 = struct('xc',4,'yc',0,'r',1,...
           'c',[.2 .5 .3]);
```

Lecture 20

21

## Example: Averaging two disks



Lecture 20

24

## Example: compute "average" of two disks

```
% D1 and D2 are disk structures.
% Average is:
r = (D1.r + D2.r) / 2;
xc = (D1.xc + D2.xc) / 2;
yc = (D1.yc + D2.yc) / 2;
c = (D1.c + D2.c) / 2;

% The average is also a disk
D = struct('xc',xc,'yc',yc,'r',r,'c',c)
```

Lecture 20

25

How do you assign to **g** the green-color component of disk **D**?

```
D = struct('xc',3.5,'yc',2,...
          'r',1.0,'c',[.4 .1 .5])
```

A: **g = D.g;**

B: **g = D.c.g;**

C: **g = D.c.2;**

D: **g = D.c(2);**

E: *other*

Lecture 20

26

A structure's field can hold a structure

```
A = MakePoint(2,3)
B = MakePoint(4,5)
L = struct('P',A,'Q',B)
```

Recall that a Point has the fields x, y

- This could be used to represent a line segment with endpoints P and Q, for instance
- Given the MakePoint function to create a point structure, what is **x** below?

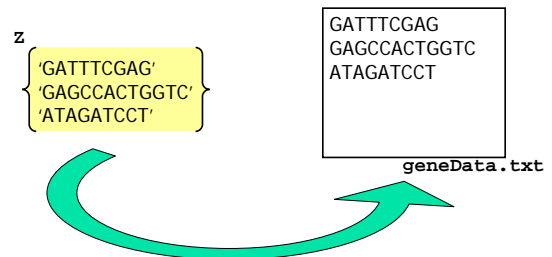
```
x = L.P.y;
```

A: 2   B: 3   C: 4   D: 5   E: error

Lecture 20

27

## Example: Write a cell array of gene sequences to a file



Lecture 20

31

### A 3-step process to read data from a file or write data to a file

1. (Create and ) **open** a file
2. **Read** data from or **write** data to the file
3. **Close** the file

Lecture 20

32

### 1. Open a file

```
fid = fopen('geneData.txt', 'w');
```

An open file has a  
file ID, here stored  
in variable **fid**

Name of the file  
(created and) opened.  
**.txt** and **.dat** are  
common file name  
extensions for plain  
text files

Built-in function  
to open a file

'w' indicates  
that the file  
is to be  
opened for  
writing

Use 'a' for  
appending

Lecture 20

33

### 2. Write (print) to the file

```
fid = fopen('geneData.txt', 'w');
```

```
for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end
```

Printing is to be  
done to the file  
with ID **fid**

Substitution sequence  
specifies the *string*  
format (followed by a  
new-line character)

The  $i^{\text{th}}$  item  
in cell array **Z**

Lecture 20

35

### 3. Close the file

```
fid = fopen('geneData.txt', 'w');
```

```
for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end
```

```
fclose(fid);
```

Lecture 20

36

```
function cellArray2file(CA, fname)
% CA is a cell array of strings.
% Create a .txt file with the name
% specified by the string fname.
% The i-th line in the file is CA{i}
```

```
fid= fopen([fname '.txt'], 'w');
for i= 1:length(CA)
    fprintf(fid, '%s\n', CA{i});
end
fclose(fid);
```

Lecture 20

37

Reverse problem: Read the data in a file line-by-line and store the results in a cell array

```
GATTCGAG
GAGCCACTGGTC
ATAGATCCT
```

geneData.txt

**Z**

```
{ 'GATTCGAG'
  'GAGCCACTGGTC'
  'ATAGATCCT' }
```

Lecture 20

38

In a file there are hidden “markers”

```
GATTTCGAG •
GAGCCACTGGTC •
ATAGATCCT •
■
```

geneData.txt

• Carriage return marks the end of a line

■ eof marks the end of a file

Lecture 20

39

Read data from a file

1. **Open** a file
2. **Read** it line-by-line until eof
3. **Close** the file

Lecture 20

40

### 1. Open the file

```
fid = fopen('geneData.txt', 'r');
```

An open file has a file ID, here stored in variable **fid**

Name of the file opened. **txt** and **dat** are common file name extensions for plain text files

Built-in function to open a file

'r' indicates that the file has been opened for reading

Lecture 20

41

### 2. Read each line and store it in cell array

```
fid = fopen('geneData.txt', 'r');
```

```
k = 0;
while ~feof(fid)
    k = k+1;
    Z{k} = fgetl(fid);
end
```

False until end-of-file is reached

Get the next line

Lecture 20

42

### 3. Close the file

```
fid = fopen('geneData.txt', 'r');
```

```
k = 0;
while ~feof(fid)
    k = k+1;
    Z{k} = fgetl(fid);
end
fclose(fid);
```

Lecture 20

43

```
function CA = file2cellArray(fname)
% fname is a string that names a .txt file
% in the current directory.
% CA is a cell array with CA{k} being the
% k-th line in the file.
```

```
fid = fopen([fname '.txt'], 'r');
k = 0;
while ~feof(fid)
    k = k+1;
    CA{k} = fgetl(fid);
end
fclose(fid);
```

Lecture 20

44